Self-organizing Prediction in Smart Grids through Delegate Multi-agent Systems

Leo Rutten[1], Paul Valckenaers[1,2]

[1]IWT department, KHLim, Diepenbeek, Belgium
{leo.rutten, paul.valckenaers}@khlim.be

[2]Mechanical Engineering Department, KU Leuven, Leuven, Belgium
Paul.valckenaers@mech.kuleuven.be

Abstract. This paper discusses a contribution to software and system engineering for smart grids, which comprises multi-agent application domain modeling and delegate multi-agent systems. In this contribution, domain models are software components – agents offering executable services – that become part of the multi-agent software as it will be deployed. These domain models crystalize relevant power engineering knowhow and expertise, thus building bridges between the power engineering and the software engineering communities. The longevity of the real-world counterparts of these domain models ensures their technical feasibility and economic value. By mirroring real-world counterparts throughout their full life cycle, re-configurability is ensured and, in combination with the evaporate-and-refresh mechanisms of the delegate multi-agent systems, even becomes business-as-usual. The paper's main contribution originates from research addressing self-organizing prediction of smart grid operations.

Keywords Delegate multi-agent systems, smart grid, self-organizing prediction, Holonic systems.

1 Introduction

Smart grid software design and development faces some unique challenges; it is indeed a highly demanding problem domain for software engineering. Conversely, translating and applying the state-of-the-art in software engineering to smart grid development is vital to cope with smart grid complexity, diversity and heterogeneity. A continued lagging of the state-of-the-art in software engineering (which unfortunately is quite common in industrial automation) would represent a significant loss to society.

A key challenge is bridging the gap between the power engineering and the software engineering communities. Jackson [1] claims that a software develop-

ment team needs to master the problem domain — in casu, power engineering — to be able to successfully develop mission-critical software systems.

In contrast, Jackson pointed out at the IBM Chair in 1985 in Leuven (B) that the problem domain is the most stable aspect in software developments. Using administrative software as a sample problem domain, Jackson rightfully mentioned that *the employee life cycle* remains largely unchanged during decades, whereas functional requirements (i.e. detailed specifications of the management reports) are likely to change almost on a weekly basis.

This observation equally applies to the smart grid. Grid components will not change overnight. The development and introduction of transformer technology, power generating technologies, power consumption devices, transmission cable materials all require significant amounts of time and effort while the technological changes rarely have an impact concerning properties that impact grid coordination and control. Likewise, the grid itself will not change overnight. Installing or replacing transformers, generators, transmission lines happens slowly from an ICT perspective.

This paper presents a conceptual design on how to capitalize on Jackson's insight and observation by emphasizing problem domain models as an important element in the software systems that are to be developed and deployed for the smart grid. Because of the presence of a multitude of stable elements in the problem domain, crystallizing power engineering knowhow and expertise within software components will be possible, both in the technical and economic sense. These durable software components would be the analogous to maps in navigation systems (note: these envisaged software models, capturing power engineering knowhow, are part of the software system when deployed).

This paper first discusses its scope: the specific problem domain that it addresses within the smart grid context. Next, it present its software system engineering approach and software mechanism (patterns) that enable to adapt to changes in the world of interest. Specifically, a delegate multi-agent system is used to collect, process and disseminate relevant information, which is "not local to a stable element in the problem domain" but needed or relevant for managing the smart grid operations. Finally, the contribution to smart grid research is discussed.

2 Scope

The research discussed in this paper is complementary to smart grid real-time control. It focuses on a time window beyond the reach of grid control systems that manage the grid operations in (hard) real time; *its time window starts from minutes into the future*. On the other hand, its time window is bounded by the build-up of uncertainty when looking farther into the future. Typically, uncertainty about wind and sun renders predictions useless beyond a number of days into the future (i.e. the software system adds nothing to the commonly available statistical infor-

mation). From a control system perspective, the envisioned software systems aims to bring and keep the grid into a comfortable state when the control performs its task. For instance, intelligent refrigerators refrain from starting their cooling when electricity supply is scarce. The envisaged software system therefore does not compete with control systems (research). Smart grids evolve in dynamic environment and so the control is done relatively to a prediction at short term.

Furthermore, the design addresses the need to anticipate (hours and days into the future) in a smart grid. Indeed, renewable energy sources dictate when they are able to supply power (e.g. wind, solar) or have complex constraints (e.g. CHP) whereas intelligent consumers are able to shift and even adapt their consumption provided that they receive relevant information ahead of time (e.g. to cool down before a peak in demand).

Importantly, the multi-agent design will predict the impact of future interactions. For instance, when e-vehicles intend/plan to start charging their batteries simultaneously, the software predicts the peak in demand, which allows the intelligent consumers to take notice of this prediction (of an undesirable future state), adapt in time and spread their load over a longer time period. This happens in a decentralized, self-organizing manner. Note that the prediction mechanism copes with situations that may never have occurred before.

Moreover, the envisaged system emphasizes precise modeling of whatever is relevant in the problem domain:

- *Full paths*: every entity on the path[1] from energy source to sink is taken into account. E.g., the system will predict the temperature of the transformers in the grid, assuming the current intentions or plans are executed.
- *All relevant attributes of the energy flows*: power, reactive power (cos Φ), balance across phases… will be covered by the prediction mechanism when and where relevant.
- *Control laws and policies*: the prediction mechanism accounts for the prevailing decision and control mechanisms in the grid. In fact, they are treated as elements belonging to the problem domain.
- *Dispatch-ability*: the ability of a consumer or producer to react to external commands or controls. This information enables a control system, for instance, to minimize the impact of a disruption by dispatching the least-effected consumers and producers. It also allows the application to monitor and manage the margins to handle disruptions dynamically.
- *Negligible discretization errors*: the system does not impose time buckets or other discretization that may impact the applicability of its models.

---

[1] This part of the research presently copes with distribution (i.e. the grid is *not meshed at any given instant in time* although grid connectivity may change over time) whereas meshed transmission networks remain to be addressed in future work. Currently, the meshed subnets within a transmission net are considered to be single copper plates.

- *Refresh and evaporate*: the system regular regenerates information, including the predictions, to account for any changes and disturbances.
- *Flexible mechanisms to collect compute and disseminate information*: the design allows application developers to add what they need (e.g. pricing for dispatching rights).

3 Self-organizing Prediction in Smart Grids

This section discusses the system architecture and design that provides a self-organizing prediction service in smart grids while handling grid re-configurations.

3.1 Structural Decomposition

The approach is centered on problem domain modeling, where these models become part of the finally deployed software. Accordingly, structural decomposition comes naturally and, as a consequence, the software system design scales with the size of the underlying grid in the real world. The main classes of problem domain entities are: Resource types; Resource instances; Activity types; Activity instances.

Typical resources are generators like wind mills, consumers like a refrigerator, power transport lines and transformers.

The resource type (model) for e.g. a power transformer mirrors its technical characteristics. An important part of this model reflects how its temperature changes in function of an electric current profile, an initial temperature and the environmental conditions. It also models the impact of this temperature on the life expectancy of this transformer.

The resource instance of such a transformer tracks the state of a physical instance. This model also comprises connections to the resource instances connected to this transformer. Resource instances (models) thus allow to discover the entire grid to which they are connected. Through a regular refresh, any reconfiguration will be detectable. Resource instances also offer a capacity reservation service, which the delegate MAS (multi-agent systems) use to generate the predictions.

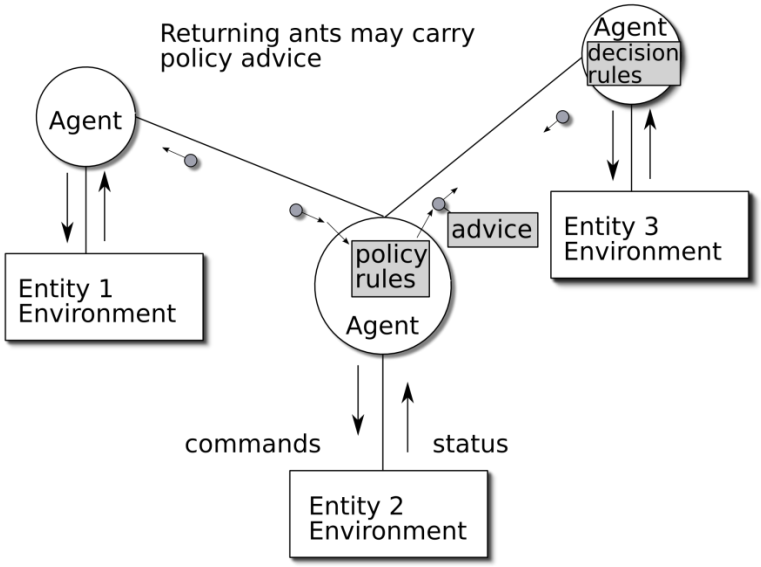

Fig. 1. Agents represent real life entities.

Activity type models reflect the manners in which tasks (e.g. ensuring that a refrigerator stays cold enough) can be performed. Note that this typically will be non-deterministic models capable of generating alternative courses of action (e.g. shifting power consumption in time).

Activity instances mirror an actual task, which includes its state but also its intentions. A refrigerating activity instance thus knows the current state (temperature, content) and a prediction of future cooling periods.

Resource and activity instances also model any policies that they apply. Indeed, decision making mechanisms are considered to reside in the domain model. E.g. a dump refrigerator will have a fixed temperature at which it switches cooling on or off. The more intelligent ones will use several delegate multi-agent systems to coordinate on a system-wide scale.

The approach makes the distinction between types and instances within the application domain, not the software.

3.2 Single Source of Truth

The above structural decomposition applies the *single source of truth principle*, which is closely related to database normalization. Importantly, whenever something changes in the world of interest, only a single model needs to be updated (preferable by an automated tracking mechanism).

This principle simplifies operations radically and eliminates many sources of errors and problems. For instance, any attempt at double bookings will be resolved by the allocation policy of the affected resource. Indeed, its resource instance model is the only source of truth concerning its state and intentions (about future states).

This approach, however, leaves an issue unresolved: how to collect, compute and deliver information that is not collocated with a domain entity (resource or activity)? How does the system predict the future states of resources and trajectories of activities, which obviously involve other resources and (interactions with) other activities? That is addressed through delegate multi-agent systems [2],[3].

3.3 Delegate MAS

A D-MAS (delegate multi-agent system) is a software pattern — implementing a mechanism to make non-local information remotely available (i.e. where the information is needed). A forget-and-refresh mechanism ensures that such information is updated regularly such that changes and disturbances are accounted for within a short delay. This pattern is a bio-inspired design — by ant colony food foraging — turning the above single-source-of-truth model of the grid into a major part of the overall solution.

A delegate MAS consists of a stream of lightweight agents called ants or ant agents, which perform an information gathering and/or dissemination task on behalf of their creator (typically a resource or activity). The delegate MAS preserves the computational efficiency of its natural inspiration (i.e. complexity of computations and communication is low-polynomial).

Without being exhaustive, every grid activity instance utilizes an exploration D-MAS and an intention-propagating D-MAS. The exploring D-MAS discovers and collects possible solutions for the task-at-hand. The intention-propagating D-MAS reserves capacity at the selected resources (i.e. full path from generator to consumer). ***These reservations make the expected future interactions (contention) visible***. Forget-and-refresh allows the activities to adapt as the situation evolves and develops. The traveling of ants is shown in fig. 1. This figure also shows how returning ants carry extra advice added by intermediate resources. Consumers use this advice to refine their decision taking rules.

3.4 Smart Grid as a Dynamic Environment

As stated above, to cope with dynamic changes and disturbances in the smart grid, information has only a limited life time. It evaporates (is forgotten) and must consequently refreshed. Resource capacity reservations are regularly reconfirmed

by intention-propagating ants. During such refresh, any changes are discovered (e.g. an overly optimistic windmill unable to deliver the promised power). Likewise, the exploration D-MAS may discover superior solutions relative to the current intentions.

As a consequence, activities may need or want to change intentions. If such change happens too easily, the predictions will suffer (as they reflect these intentions). If such change happens too rarely, system performance suffers when opportunities are lost and disturbances are not addressed. Hadeli [4] has investigated this matter and proposed mechanisms to balance responsiveness against stability. Already reserved capacity has some stickiness and changing intentions is submitted to a stochastic decision process preventing disturbances from triggering an avalanche or stampede to alternative solutions. Hadeli's investigations show a large sweet spot: a modest amount of stickiness or a modest willingness to change deliver proper behaviors.

4. Software Engineering for Smart Grids

This paper addresses the challenge of building bridges, in a figurative sense, between the power engineering and the software engineering domains. It comprises the elaboration of executable software models that capture the relevant domain knowhow and expertise to address this challenge in a durable manner. The envisioned software systems do not longer rely exclusively on specialized — in the application domain —human development teams (cf. Jackson in [1]) to account for the specific nature of the smart grid domain. And, the resulting systems are able to account for power grid concerns that are ignored elsewhere, such as by electronic markets matching producer and consumer without accounting for transmission or distribution.

The envisioned solution includes the delegate MAS pattern allowing to use a single-source-of-truth design. Single-source-of-truth permits to extend every entity in the grid with a software extension as its only point of reference whilst the non-local information is generated and refreshed by the appropriate delegate MAS. This is analogous to a map technology where the map also predicts traffic (congestion) by the delegate MAS projecting user intentions (note: FP7 Project MODUM is developing such intelligent traffic application using D-MAS).

For future research in software engineering for the smart grid, this implies the elaboration of executable domain models and delegate MAS variants. This is analogous to a map technology where the map's legend needs to be filled with all elements occurring in the world-of-interest. In fact, the research aims to build a base, mirroring the power grid, on which applications are realized and maintained swiftly and with little effort, analogous to navigation on top of a suitable map. Another analogy is to consider the domain models and delegate MAS as a 'grid

operating system' responsible for a predictive resource allocation layer on top of which applications execute.

Furthermore, the envisaged systems are compliant with emerging information technologies, largely thanks to their structural decomposition. In particular, the prototype implementation in OTP/Erlang is compliant with high performance computing (but not GPU) and cloud computing. Note that OTP/Erlang supports software updating at run-time and allows to achieve extreme levels of service availability (routinely up to 99.999% and even better with extra effort).

5. Conclusion

5.1 Contribution

This paper discusses how to build bridges between the power engineering and the software engineering communities. It applies a core achievement in software engineering — domain modeling — and discloses the extent of what can be achieved with this approach in smart grids:

- Single source of truth designs capable of collocating their problem domain models with their real-world counterparts, offering their services inside deployed software systems. Thus, the models may accompany their real-world counterparts where a single model can be reused wherever its domain entity exists.
- A delegate MAS pattern to collect, compute and disseminate relevant information that cannot be collocated with a long-lived domain entity.
- Delegate MAS combined with domain model services that generate, and regularly update, predictions of resource states and activity trajectories accounting for properties and attributes as needed. Among others, power transport and transformation between generator and consumer are accounted for.

Overall, the research aims at and enables to capture power engineering expertise into software, complementing expertise in brains of software developers (which remains necessary nonetheless).

5.2 Future Research

The current research results cover only a small subset of the problem domain models needed to cope with and contribute to smart grid operations. The base mechanisms have been established but the collection of executable models is in-

complete. Moreover, one important problem domain aspect remains future research: meshed power networks. The base architecture puts forward staff holons whenever a non-local concern needs addressing. The smart grid needs such a holon that computes the available capacities in between connection points of a meshed network. This staff holons needs to recompute and refresh this information regularly, possibly shifting from optimistic estimates far ahead in the prediction time window toward conservative bounds when approaching the instant where real-time control takes over.

Finally, the delegate MAS pattern needs to be applied to building up commitments gradually within the time window. Farther in the future, commitments are weak while exploration dominates. Closer to the present, these commitments have to be firm. In the presence of multiple independent organizations, commitments involve pricing and contracts. Pricing needs to induce proper behavior (i.e. prevent cheating and manipulation). For a gradual commitment buildup, pricing mechanisms need to cover options, cancelations, dispatch-ability rights, etc. Here, market deregulation will be key.

References

1. M. Hinchey, M. Jackson, P. Cousot, B. Cook, J. P. Bowen, and T. Margaria, “Software engineering and formal methods,” Commun. ACM, vol. 51, no. 9, pp. 54–59, 2008.
2. T. Holvoet, D. Weyns, and P. Valckenaers, “Patterns of delegate mas,” in SASO. IEEE Computer Society, 2009, pp. 1–9.
3. H. V. D. Parunak, S. Brueckner, D. Weyns, T. Holvoet, P. Verstraete, and P. Valckenaers, “E pluribus unum: Polyagent and delegate mas architectures,” in MABS, ser. Lecture Notes in Computer Science, L. Antunes, M. Paolucci, and E. Norling, Eds., vol. 5003. Springer, 2007, pp. 36–51.
4. Hadeli, “Bio-inspired multi-agent manufacturing control systems with social behaviour,” Ph.D. dissertation, KULeuven, Faculty of Engineering, 2006.